Robotics

Control I

Admin

- quiz
- any questions about project or otherwise?

Control

- Now we have robot parts
  - sensors
  - actuators
  - now need to make it do stuff
  - need control
  - simplest control
    - feedback

Feedback

- Feedback control
  - getting robot to achieve/maintain goal by comparing current state to goal state.
  - Feedback:
    - information “fed back” or input into robot
    - sensor data
  - goal state:
    - as with AI this is the desired state

Goals

- Achievement Goals:
  - robot has done it, its done
- Maintenance goals:
  - as long as goal is being met, we are on track
  - your current lab?

Error

- Error
  - according to control theory
  - difference from desired state to current state
- binary error:
  - in goal state
  - out of goal state
  - useful?

Error II

- Direction of error
    - current lab, might care about direction of error
    - off to left or off to right
- Magnitude of error
    - sometimes get magnitude info too
        - hot and cold game example
        - sensor readings of walls.

Wall following robot example

- feed back control
  - task: follow right hand wall at approx 6 inches from wall
  - discuss

Oscillation

- Depending on the magnitude of the correction
  - can oscillate a little or a lot
  - want to decrease this.

Proportional control

- Simple feedback control
  - $o = K_p i$
  - where
    - o : output
    - i: is input (error)
    - K_p : is a proportionality constant
  - eg: wall follower
    - error positive or negative
    - want to correct
  - K_p is usually arrived at by trial and error
  - Ahem – that is “empirical testing”

Damping

- proportional control still has oscillations
    - Damping:
        - systematically decreasing oscillations
        - with proportional control depends on picking a good proportionality constant.

Derivative control

- next derivative control attempts to compensate for momentum (aka predict the future)
  - momentum = mass*velocity
  - as system gets closer to goal subtract amount proportional to velocity
  - $o = K_d \, (di/dt)$
  - di = change in error
  - dt = change in time
  - K_d = proportionality constant.
- Not used by itself
- Susceptible to error spikes

PD Control

- proportional Derivative control
  - $o = K_p i + K_d (di/dt)$
  - sum of two gives better control
    - industrial process control
  - According to industrial sites, often used for servo control

integral control

- integral control
  - system keeps track of (sums up) own errors
  - tries to minimize steady state (repeated) errors
  - $o = K_f \int (i(t)*dt)$
  - i(t) error at time t
  - dt change in time since last
  - K_f : new constant

So what does that mean

- We are summing up the the error for all time.

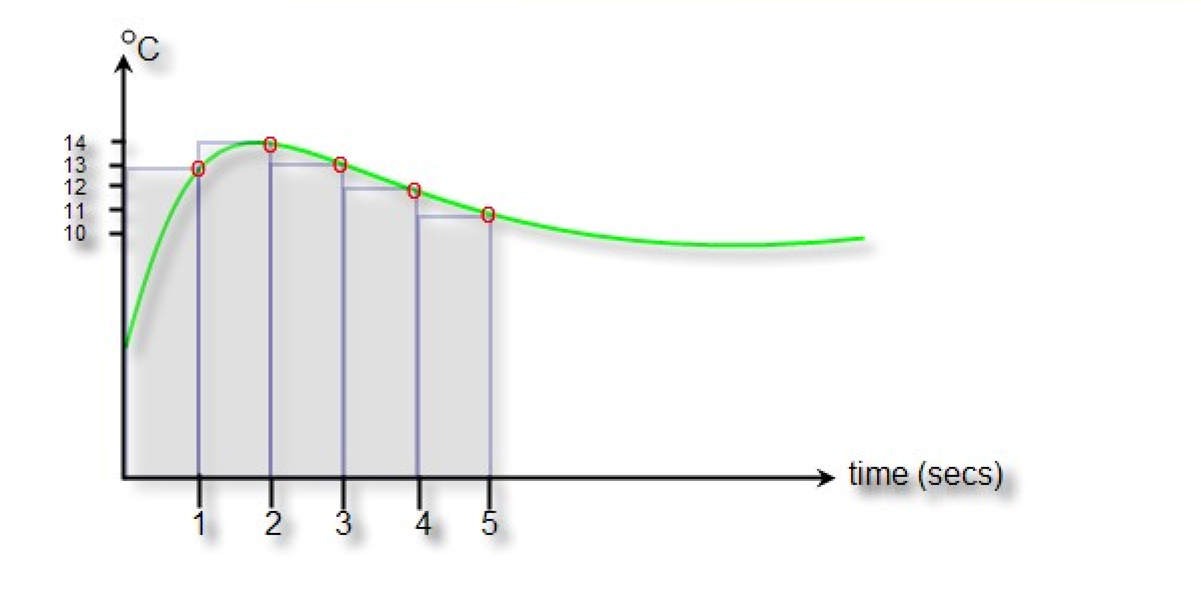

Image credit
https://www.csimn.com/CSI_pages/PIDforDummies.html

- While this curve shows positive error, more likely sometimes error will be negative
- Allows keeping track of a long term history in a single variable.

Implementing Integral Control

- Implementation is actually much easier than it looks
  - Assume a variable int_error that persists from one loop run to the other.
  - int_error = int_error + current_error*time_since_last_read
  -

PI Controller

- PI controllers are really common in industrial applications
  - Uses instantaneous error P
  - Historical error I
  - o= $K_p i + K_f \int i(t)*dt$
  - Fantastic quiz2 question: give me the python code for a pi controller.

PID control

- sum up proportional, integral, and derivative control
    - add accuracy
    - o= $K_p i + K_f \int i(t)*dt + K_d\ (di/dt)$
    - of course you might have to tweak the gains
        - the K terms

feedback control

- using input/error/ world state “fed back” into the robot to help robot accomplish goal

feed forward control

- in this, no sense data used
- just world models
- good for?

Reading

- For basic control theory for non-engineers
  - https://www.csimn.com/CSI_pages/PIDforDummies.html
  - An example PID controller in python
    - http://code.activestate.com/recipes/577231-discrete-pid-controller/
    - Note this code seems to assume that all delta-T values are 1.
    - Great in a perfect world